Technical Report

IBM Platform Load-Sharing Facility (LSF) Integration with NetApp Storage

An Implementation and Configuration Guide

Bikash Roy Choudhury, NetApp
October 2013 | TR-4237

Abstract

IBM Platform Load-Sharing Facility (LSF) is a popular batch job scheduler used in large compute farms in various engineering, chip design, and manufacturing environments. This paper provides information about how to implement and configure the new LSF storage-aware plug-in on NetApp® storage. The storage-aware plug-in from NetApp monitors and reports the scheduler about the available storage resources for the jobs submitted in the compute farm. It also identifies and notifies the administrator of any hot jobs running in the compute farm. The LSF scheduler with the help of the storage-aware plug-in can now take informed decisions while submitting jobs in the compute farm to reduce job failures.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Platform Computing, now part of the IBM, Load-Sharing Facility (LSF) tool, is the most commonly used job scheduler in engineering design, chip manufacturing, and other high-performance computing applications. LSF provides features and functionalities to schedule, pend, suspend, and execute jobs submitted by users through different queues. However, with the growing complexity in the design and the growing number of cores in compute farms, the number of concurrent jobs is increasing day by day.

The back-end storage that stores and processes the jobs frequently runs out of resources as the number of jobs in the compute farm increases. Recent industry data indicated that there is a high rate of the job failures due to lack of storage space and resources. Jobs frequently fail when they compete for overutilized storage resources. Having to resubmit failed jobs extends the design cycle timeline and thus affects the time to market (TTM).

NetApp has, for the first time, come up with a plug-in that provides information to LSF about the storage resources available, identifies hot or runaway jobs, and also collects historical I/O information. NetApp is providing an LSF scheduler plug-in to make decisions to run or pend jobs based on storage resource information. The NetApp scheduler plug-in allows the LSF scheduler to make an informed decision while submitting jobs to the compute farm.

2 Intended Audience

This guide will help LSF and storage administrators to configure and deploy the NetApp plug-ins and other dependencies to make the LSF scheduler more “storage-aware.”

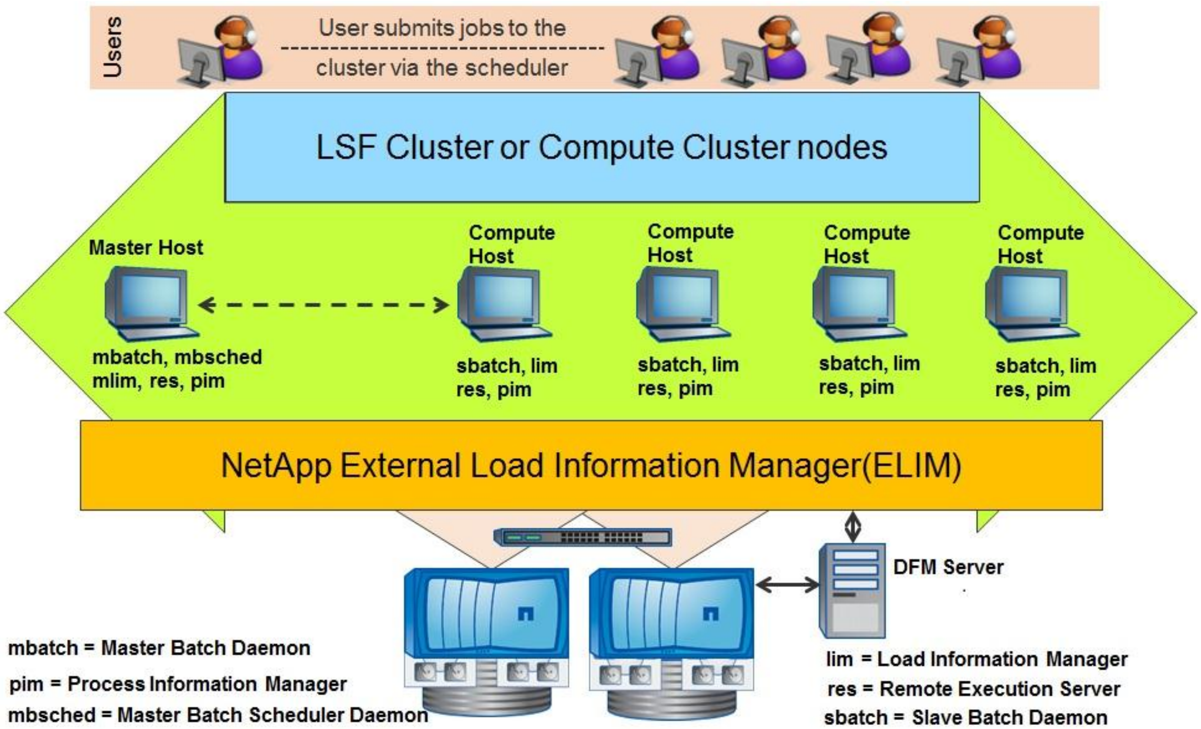
3 LSF Architecture

LSF consists of a master node and a number of execution hosts. The master node would run the following daemons: mbatch, mbsched, mlim, res, and pim. The execution hosts will have the sbatch, lim, res, and pim running on them.

- The **mbatch** is the master batch daemon that runs on the master node and is responsible for the overall state of all the jobs running in the compute farm.
- The **mbsched** is the master batch scheduler daemon that runs along with the mbatch on the master node to schedule the jobs based on the resources available and job requirements.
- The **mlim** is the master load information manager that runs on the master node and is responsible for gathering load information from the LIMs that are running on other execution hosts in the compute farm.
- The **res** is the remote execution server that runs on the master node and all the other execution hosts that is responsible for executing the jobs that are submitted by the users.
- The **pim** is the process information manager that runs on all the nodes in the compute farm, including the master. This is responsible for gathering information about the CPU and memory and other utilization of resources for the jobs running.
- The **sbatch** is a slave batch daemon that runs on each of the execution hosts in the compute farm other than the master node. The sbatch forks a child sbatch daemon that works along with res on each job running on every node in the compute farm. When the jobs complete, the child sbatch process exists on each execution node. The master node can also be an execution node.
- The **elim** is an external load information manager and is a user-defined script that is responsible for tracking and gathering the load information from an external plug-in. This provides the information to pim that passes on to the sbatch running on each execution host.

(Source: http://support.sas.com/rnd/scalability/platform/PSS5.1/lsf7.05_foundations.pdf.)

Figure 2) LSF Architecture with NetApp Plugin integration



4 NetApp Plug-In Implementation

NetApp has developed an external plug-in that helps the LSF scheduler gather information about the available storage resources. The NetApp LSF scheduler plug-in framework does not communicate with the LSF scheduler directly using any APIs. The plug-in presents itself with the storage resource information to the LSF external module. The LSF scheduler is responsible to query this external scheduler plug-in periodically as scheduled to gather information on the storage resources. The plug-in is designed to perform the following functions:

- Historical job I/O logging
- Hot job detection
- “Storage-aware” job scheduling

The NetApp LSF plug-in consists of two parts. Both of them share the information polled from the DataFabric® Manager (DFM) or Operations Manager to identify the storage resource utilization and also identify hot jobs that are using up the storage resources.

- NetApp scheduler plug-in
- NetApp compute agent with hot job detector plug-in

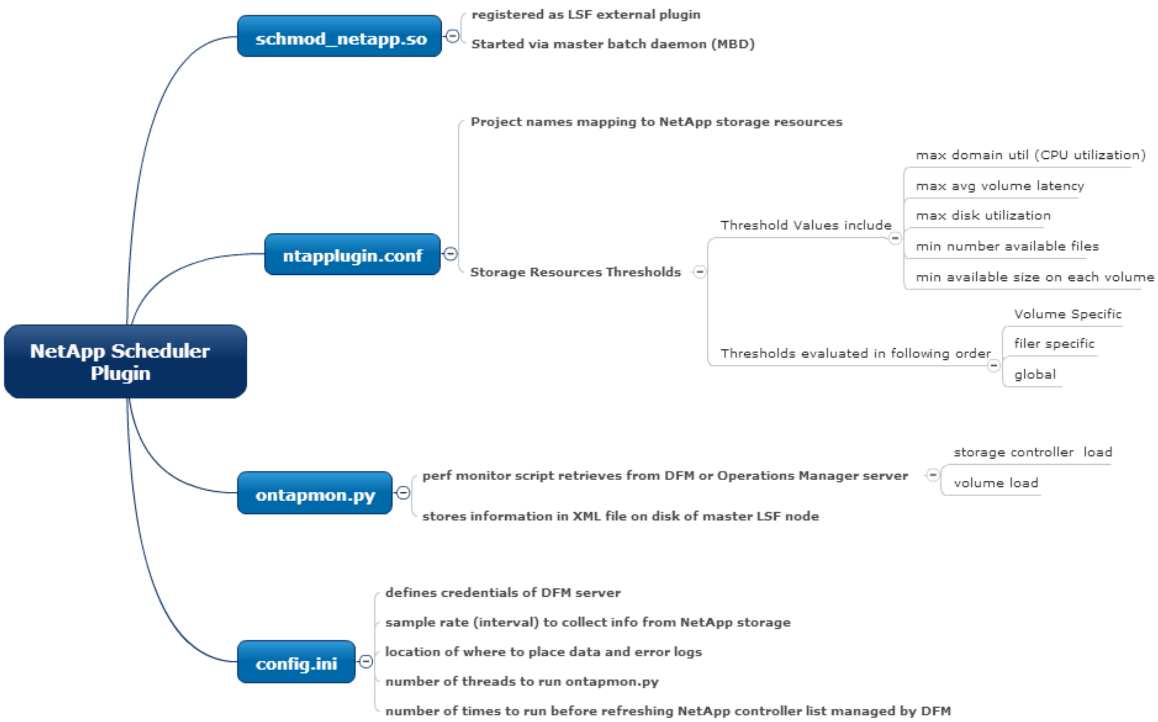
Note: The NetApp compute agent with hot job detection requires specific setup and installation work prior to use. This setup work might be difficult for some customers with a large number of compute nodes. Section 7 in this document lists the prerequisites for NetApp compute agent with hot job detection to work properly. However, the NetApp scheduler plug-in can function independently and be installed separately from the NetApp compute agent with hot job detection.

The NetApp LSF plug-in consists of the NetApp scheduler plug-in and NetApp compute agent with hot job detection modules that would retrieve NetApp storage and load information from the DFM server.

4.1 NetApp Scheduler Plug-In

The NetApp scheduler plug-in consists of the following:

Figure 2) Tree representation of NetApp scheduler plug-in



- **schmod_netapp.so**. Provided as a set of C source files and built on the master LSF node to generate the plug-in (`schmod_netapp.so`) file. The module is registered as an LSF external plug-in and started using the master batch daemon (MBD). The purpose of the external LSF scheduler plug-in is to look into the performance data collected by the storage monitoring script (ontapmon.py) and then compare the latest performance data against the thresholds to determine if the job should be allowed to run.
- **ntapplugin.conf**. This configuration file contains information on:
  - Project names mapping to NetApp storage resources
  - Threshold values for storage resources

  Threshold values can be set globally or specific to a storage controller or volume. These threshold values include:

  - Maximum domain utilization, which provides CPU utilization
  - Maximum average volume latency
  - Maximum disk utilization
  - Minimum number of available files
  - Minimum available size on each volume

  The thresholds are evaluated in the following order:

  1. Volume specific
  2. Filer specific

3. Global

   For instance, if volume-specific thresholds are defined, then it will compare values with volume-specific thresholds and ignore other filer and global thresholds.

- **ontapmon.py**. This is the performance monitoring script from NetApp that retrieves storage controller and volume load information from DFM or the Operations Manager server. It stores the information in an XML file on disk of the master LSF node.
- config.ini. This is the configuration file for the ontapmon.py script that defines the credentials of the DFM server, the sample rate (interval) to collect information from NetApp storage, location of where to place data collected and error logs, number of threads to use to run ontapmon.py script, and the number of times to run before refreshing the NetApp controller list being managed by the DFM.

4.2 NetApp Compute Agent with Hot Job Detection Plug-In

This module logs historical NFS I/O data generated by LSF jobs running on compute nodes. The `netapp_lsf_hot_job_detector.py` script uses threshold values to identify performance problems and the hot or runaway LSF jobs that are performing the most operations on the affected volumes. This data is sent to an administrator in an e-mail alert.

The NetApp compute agent with hot job detection plug-in consists of:

Figure 3) Tree representation of hot job detection plug-in

- **elim.netapp_compute**. This is responsible for monitoring and logging the NFS read and write operations performed by every job that runs in the compute farm with the help of `SystemTap` on each of the nodes in the compute farm. It generates job reports for every single job running on each node. For each LSF job, a text file is written listing the number of NFS reads and writes performed by that job. If 100 jobs are running, then there will be 100 job reports. Job report files are written to a shared directory on a configurable interval (default value of 30 seconds).This can be modified in the `netapp_lsf_compute_agen.conf` file.
- The script is stored in a shared location on NetApp storage, where all of the compute nodes can access it.
- **netapp_lsf_hot_job_detector.py**. This Python script is responsible for monitoring Data ONTAP® performance by reading the XML files written by the `ontapmon.py` performance monitoring script. Performance data in these files is compared against thresholds set in this script's configuration file (`netapp_lsf_hot_job_detector.conf`). When a performance problem is found, this script will

search the LSF job report text files written by the `elim.netapp_compute` to determine which LSF jobs have performed the most recent operations against the affected storage controller and/or volume. The report is written in the job directory location where the rest of the text files are present.

- **netapp_lsf_hot_job_detector.conf**. This is the configuration file for the hot job detector script. Performance threshold values are set here to determine when the hot job detector should issue alerts. These threshold values could be same as or different from the one set in the `ntapplugin.conf` in the NetApp LSF plug-in. For each performance problem, this script will look for the LSF jobs that have performed the most recent operations against the affected volume. These top jobs are included in the performance problem report that can be stored in a different location on the NetApp storage.
- **netapp_lsf_hot_job_email.py**. This is a Python script that notifies administrators through e-mail messages, alerting them about any performance problems detected and listing those LSF jobs that have performed the most operations on the affected storage controller and volume. After the performance problem is identified by the `netapp_lsf_hot_job_detector.py`, it internally calls the `netapp_lsf_hot_job_email.py` script to send an e-mail notification.

5 NetApp LSF Plug-In Workflow

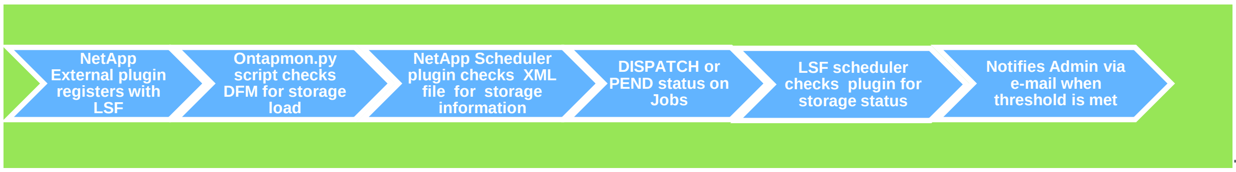
The NetApp LSF scheduler, after being set up and configured correctly, will perform the desired functions as listed in section 4.1. The following section describes the workflow of the NetApp plug-in, including gathering storage resource performance data, checking threshold values, and reporting any performance issues. Figure 4 shows the NetApp scheduler plug-in workflow.

5.1 NetApp Scheduler Plugin

- The `schmod_netapp.so` module is first added to the `lsb.modules` file. Restarting the master batch daemon (MBD) running on the LSF master node registers the external plug-in to the LSF scheduler. After the module is registered and started, it checks the `ntapplugin.conf` for the project names that map to storage resources, thresholds, working directories, and debug modes. The global, storage, and volume-specific threshold values are set in the configuration file.
- The `ontapmon.py` script polls the DFM server periodically (by default 60 seconds) to get the most recent storage resource statistics based on the preceding listed variables. This monitoring script can probe the DFM server sooner than 60 seconds. It is not recommended to set the polling interval to be more frequent than 30 seconds.
- The information from the DFM server is saved in a `<NetApp_controller_hostname>.xml` file in the counter directory as configured in the `ntapplugin.conf` and `config.ini` files. In this paper the counter directory location will be referred to as `/var/log/plugin/DIRLOC`.
- The storage controller performance .xml files are updated by `ontapmon.py` every 60 seconds by default. The `schmod_netapp.so` module checks the updated values in the XML file and compares them to the threshold values set in the `ntapplugin.conf` file.
  - After the resource utilization goes beyond the threshold values, the plug-in sets a `PEND` status on jobs that will target one or more of the affected volumes. If all volumes in the groups targeted by the job are free of performance problems, the plug-in will set a `DISPATCH` status on the job. The status can be verified in the `ntapplugin.log` file in the working directory as configured in the `ntapplugin.conf` file. In this paper the working directory is referred to as `/var/log/plugin/netapp-log`.
  - The global values always take precedence over the local threshold values. This means that if a single aggregate on a NetApp storage controller has multiple volumes and any of the global threshold values are met from any one volume, then the scheduler sets a `PEND` status on the rest of the jobs in the queue. Likewise, the NetApp LSF Scheduler would also set a `PEND` state on the jobs in the queue if a specific volume has met the local threshold values set in the `ntapplugin.conf` file.

- The LSF scheduler will periodically call the NetApp scheduler plug-in to ask if a job that is in a `PEND` state can now be allowed to `RUN`. This interval is configurable in the LSF options. NetApp LSF plug-in does not have any control over the information passing as it does not report to LSF scheduler.
- After the plug-in sets the status to PEND on the existing jobs in the queue, an e-mail notification goes out to the storage admin as configured in the `netapp_lsf_hot_job_email.py` file.

Figure 4) NetApp LSF plug-in Workflow

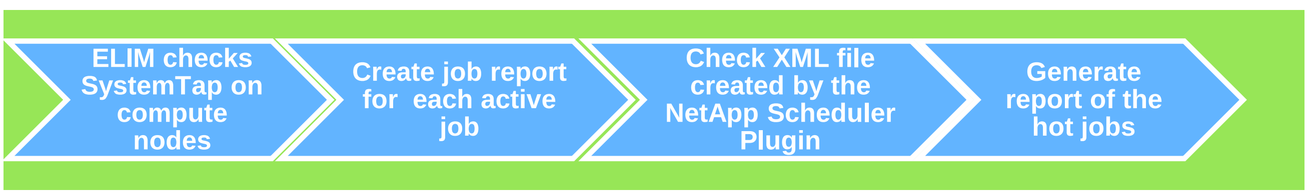


5.2 Compute Agent and Hot Job Detector Plug-In

- The compute agent plug-in relies on `elim.netapp_compute` to monitor the `SystemTap` data on each compute node. A text file is created for each LSF job and saved in a shared `job_report` directory.
  - The job report text files list the number of NFS read and write operations on each storage controller and volume. The data is separated by timestamps to indicate when the listed operations occurred.

    The `job_report` directory is configured in the `netapp_lsf_hot_job_detector.conf` file and is recommended to be configured on the NetApp shared location so that all the compute nodes can write the text files for each job that is running to one location.
- If any jobs get to the PEND state due to the storage resource utilization reasons, the hot job detector will generate the report if its configured thresholds are exceeded. By default the report will have the top three jobs, but this can be configured to list more than three jobs. Figure 5 illustrates the workflow of the hot job detection.

Figure 5) NetApp Compute Agent and Hot Job detection plug-in Workflow



6 Dependencies

The NetApp LSF plug-in requires a few dependencies in the compute environment for the plug-in to gather the storage resource utilization information and notify the storage administrator in case of any storage performance issues.

- **OnCommand 3.2**. The DFM server that is part of the OnCommand® 3.2 suite gathers the resource utilization information for the storage controllers to which it is connected. The `ontapmon.py` script from the plug-in will query the DFM server to gather information to match with the threshold values that are set in the `ntapplugin.conf` file.
- **Simple Network Time Protocol (SNTP)**. An SNTP server such as NTP is required to synchronize the time setting between the DFM server and the storage controller and reduce the clock skew between the two. More details can be found in this knowledgebase article: https://kb.netapp.com/support/index?page=content&actp=LIST&id=S:1012660.

7 Prerequisites to Install NetApp LSF Plug-In

The NetApp LSF plug-in requires certain prerequisites in the EDA environment where compute nodes use LSF to schedule the jobs that run on design files that are stored on NetApp storage. The following requirements have to be met before configuring the NetApp LSF plug-in:

- LSF8.0 is the minimum version required for using the NetApp LSF plug-in. However, beta customers tested this plug-in also with LSF 7.0 and LSF9.0. Both the LSF versions works fine with the NetApp LSF scheduler plug-in.
- The plug-in requires Linux® operating system on the compute nodes with storage mounted over NFSv3.
- RHEL6.x or SLES 11SP2 is the recommended kernel on all the compute nodes. If RHEL5.7 or a later kernel is used on the compute nodes, Python2.6 is a requirement.
- Complete `SystemTap` module, including "`-devel-`" and "`-debuginfo-`" packages, is required for the `elim.netapp_compute` to gather the NFS read and write information for each job. These packages are only required on one node per kernel version to compile the `SystemTap .ko` file.
  - The complete `SystemTap` packages are only required on one node per kernel. The rest of the compute nodes only require `SystemTap runtime`.
  - If `SystemTap` and/or `Python 2.6` cannot be installed on the compute nodes, the NetApp compute agent with hot job detection plug-in will not be able to produce job report files or identify hot jobs. The NetApp scheduler plug-in, however, does not require SystemTap and can be installed independently.
- A compiler on the Linux kernel to compile the `schmod.netapp.so` module that registers with the LSF scheduler as an external plug-in.
- Download and install the latest NetApp Manageability Software Development Kit (NMSDK) from the NetApp developer community at https://communities.netapp.com/docs/DOC-1152 in the `LSF_TOP` directory location.
- Data ONTAP 8.1.2 and later 7-Mode is the recommended version on the storage.
  - NetApp LSF plug-in is currently not supported on clustered Data ONTAP.
  - Set `options httpd.admin.enable` on.

    This option enables HTTP and XML transport paths, which help to communicate with the filer using DFM server and NMSDK.

8 NetApp LSF Plug-In: Setup and Configuration

The NetApp LSF plug-in tar ball includes the NetApp scheduler and compute agent with hot job detection plug-in. It is recommended to untar the plug-ins in the `LSF_TOP` directory. The `LSF_TOP` entry should be in the `install.config` file, which is in the LSF install directory. In this paper, the `LSF_TOP=/mnt/lsf-root/share/lsf`.

8.1 Installing and Setting Up SystemTap on Compute Node

`SystemTap` must be installed on each compute node to gather NFS operation data for LSF jobs. Installing `SystemTap` might not be required if the customer chooses not to use the NetApp compute agent and hot job detection plug-in.

1. Enable the `rhel-debuginfo.repo` file to download the required files during the `yum install` process.

```
[root@ibmx3650-svl43 ~]# cd /etc/yum.repos.d/
[root@ibmx3650-svl43 yum.repos.d]#
[root@ibmx3650-svl43 yum.repos.d]# vi rhel-debuginfo.repo
[rhel-debuginfo]
name=Red Hat Enterprise Linux $releasever - $basearch - Debug
```

```
baseurl=ftp://ftp.redhat.com/pub/redhat/linux/enterprise/$releasever/en/os/$basearch/
Debuginfo/
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

2. Check the Linux kernel version.

```
[root@ibmx3650-svl43 yum.repos.d]# uname -r
2.6.18-238.el5
```

3. Perform a `yum install` on the "`-devel-`" packages.

```
[root@ibmx3650-svl43 yum.repos.d]# yum install kernel-devel-2.6.18-238.el5
Loaded plugins: rhnplugin, security
rhel-debuginfo
| 1.2 kB     00:00
rhel-debuginfo/primary
| 919 kB     00:00
rhel-debuginfo: [################################################                ]
1745/6073

[root@ibmx3650-svl43 yum.repos.d]# yum install kernel-devel-2.6.18-238.el5
Loaded plugins: rhnplugin, security
rhel-debuginfo
| 1.2 kB     00:00
rhel-debuginfo/primary
| 919 kB     00:00
rhel-debuginfo
6073/6073
Setting up Install Process
Package kernel-devel-2.6.18-238.el5.x86_64 already installed and latest version
Nothing to do
[root@ibmx3650-svl43 yum.repos.d]#

[root@ibmx3650-svl43 yum.repos.d]# yum install yum-utils
Loaded plugins: rhnplugin, security
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package yum-utils.noarch 0:1.1.16-21.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=========================================================================================
=========================================================================================
==================
 Package                                     Arch
Version                                          Repository
Size
=========================================================================================
=========================================================================================
==================
Installing:
 yum-utils                                   noarch
1.1.16-21.el5                                    rhel-x86_64-server-5
73 k
```

```
Transaction Summary
==========================================================================================
==========================================================================================
==================
Install       1 Package(s)
Upgrade       0 Package(s)

Total download size: 73 k
Is this ok [y/N]:
Is this ok [y/N]: y
Downloading Packages:
yum-utils-1.1.16-21.el5.noarch.rpm
|  73 kB     00:00
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : yum-utils
1/1
Running Transaction
  Installing     : yum-utils
1/1

Installed:
  yum-utils.noarch 0:1.1.16-21.el5

Complete!
[root@ibmx3650-svl43 yum.repos.d]#
```

4. Perform a `yum install` on the "`-develinfo-`" package.

```
[root@ibmx3650-svl42 yum.repos.d]# which debuginfo-install
/usr/bin/debuginfo-install


[root@ibmx3650-svl43 yum.repos.d]# /usr/bin/debuginfo-install kernel-devel-2.6.18-238.el5
Loaded plugins: rhnplugin
--> Running transaction check
---> Package kernel-debuginfo.x86_64 0:2.6.18-238.el5 set to be updated
--> Processing Dependency: kernel-debuginfo-common-x86_64 = 2.6.18-238.el5 for package:
kernel-debuginfo
--> Running transaction check
---> Package kernel-debuginfo-common.x86_64 0:2.6.18-238.el5 set to be updated
--> Finished Dependency Resolution

==========================================================================================
==========================================================================================
==================
 Package                                                 Arch
Version                                        Repository
Size
==========================================================================================
```

```
=========================================================================================
==================
Installing:
 kernel-debuginfo                                        x86_64
2.6.18-238.el5                                 rhel-debuginfo
176 M
Installing for dependencies:
 kernel-debuginfo-common                                 x86_64
2.6.18-238.el5                                 rhel-debuginfo
32 M

Transaction Summary
=========================================================================================
=========================================================================================
==================
Install       2 Package(s)
Upgrade       0 Package(s)

Total download size: 208 M
Is this ok [y/N]:
Downloading Packages:
(1/2): kernel-debuginfo-common-2.6.18-238.el5.x86_64.rpm
|  32 MB     00:13
(2/2): kernel-debuginfo-2.6.18-238.el5.x86_64.rpm                                (40%)
29% [====================-            ] 2.4 MB/s |  52 MB     00:51 ETA

Downloading Packages:
(1/2): kernel-debuginfo-common-2.6.18-238.el5.x86_64.rpm
|  32 MB     00:13
(2/2): kernel-debuginfo-2.6.18-238.el5.x86_64.rpm
| 176 MB     01:09
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
------------------
Total
2.4 MB/s | 208 MB     01:26
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : kernel-debuginfo-common
1/2
  Installing     : kernel-debuginfo
2/2

Installed:
  kernel-debuginfo.x86_64 0:2.6.18-238.el5

Dependency Installed:
  kernel-debuginfo-common.x86_64 0:2.6.18-238.el5

[root@ibmx3650-svl43 yum.repos.d]#
```

5. Check if SystemTap is running on the node.

```
[root@ibmx3650-svl43 yum.repos.d]# stap -v -e 'probe vfs.read {printf("read
performed\n"); exit()}'
Pass 1: parsed user script and 76 library script(s) using 147596virt/22524res/3016shr
kb, in 140usr/10sys/145real ms.
Pass 2: analyzed script: 1 probe(s), 21 function(s), 3 embed(s), 1 global(s) using
263332virt/78764res/6148shr kb, in 830usr/200sys/3043real ms.
Pass 3: translated to C into
"/tmp/stapfa4Pnf/stap_80b71ff7ff35fba875c2e67efc8b74d9_10015_src.c" using
256216virt/77100res/7512shr kb, in 240usr/40sys/288real ms.
Pass 4: compiled C into "stap_80b71ff7ff35fba875c2e67efc8b74d9_10015.ko" in
3180usr/420sys/4202real ms.
Pass 5: starting run.
read performed
Pass 5: run completed in 10usr/70sys/594real ms.

=====================================================
```

8.2 Install and Set Up Python2.6

Python2.6 install on the compute nodes would not be required if RHEL6.x or a later kernel is used in the compute nodes. RHEL5.7 and later do require this install.

1. Install the prerequisite RPM that is not specific to any specific architecture for Python2.6 to run.

```
[root@ibmx3650-svl42 log]# wget
http://dl.iuscommunity.org/pub/ius/stable/Redhat/5/x86_64/ius-release-1.0-
10.ius.el5.noarch.rpm
--2013-01-08 14:43:07--  http://dl.iuscommunity.org/pub/ius/stable/Redhat/5/x86_64/ius-
release-1.0-10.ius.el5.noarch.rpm
Resolving dl.iuscommunity.org... 50.57.54.209
Connecting to dl.iuscommunity.org|50.57.54.209|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7331 (7.2K) [application/x-rpm]
Saving to: `ius-release-1.0-10.ius.el5.noarch.rpm'

100%[==================================================================================
====================================================================>] 7,331       --.-
K/s   in 0s

2013-01-08 14:43:17 (18.1 MB/s) - `ius-release-1.0-10.ius.el5.noarch.rpm' saved
[7331/7331]

root@ibmx3650-svl42 log]# wget
http://dl.iuscommunity.org/pub/ius/stable/Redhat/5/x86_64/epel-release-5-4.noarch.rpm
--2013-01-08 14:45:15--  http://dl.iuscommunity.org/pub/ius/stable/Redhat/5/x86_64/epel-
release-5-4.noarch.rpm
Resolving dl.iuscommunity.org... 50.57.54.209
Connecting to dl.iuscommunity.org|50.57.54.209|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 12275 (12K) [application/x-rpm]
Saving to: `epel-release-5-4.noarch.rpm'
```

```
100%[=========================================================================================
=====================================================================>] 12,275      --.-
K/s   in 0.001s

2013-01-08 14:45:15 (16.1 MB/s) - `epel-release-5-4.noarch.rpm' saved [12275/12275]

[root@ibmx3650-svl42 log]# yum localinstall epel-release-5-4.noarch.rpm ius-release-1.0-
10.ius.el5.noarch.rpm --nogpgcheck
Setting up Local Package Process
Examining epel-release-5-4.noarch.rpm: epel-release-5-4.noarch
Marking epel-release-5-4.noarch.rpm to be installed
Examining ius-release-1.0-10.ius.el5.noarch.rpm: ius-release-1.0-10.ius.el5.noarch
Marking ius-release-1.0-10.ius.el5.noarch.rpm to be installed
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:5-4 set to be updated
---> Package ius-release.noarch 0:1.0-10.ius.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

=========================================================================================
=========================================================================================
==================
 Package                                 Arch                                 Version
Repository                                                     Size
=========================================================================================
=========================================================================================
==================
Installing:
 epel-release                            noarch                               5-4
/epel-release-5-4.noarch                                       22 k
 ius-release                             noarch                               1.0-
10.ius.el5                               /ius-release-1.0-10.ius.el5.noarch
8.3 k

Transaction Summary
=========================================================================================
=========================================================================================
==================
Install       2 Package(s)
Upgrade       0 Package(s)

Total size: 30 k
Is this ok [y/N]:
Downloading Packages:
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : epel-release
1/2
```

```
  Installing     : ius-release
2/2

Installed:
  epel-release.noarch 0:5-4
ius-release.noarch 0:1.0-10.ius.el5

Complete!
[root@ibmx3650-svl42 log]#
```

2. Install `python26` using `yum install.`

```
[root@ibmx3650-svl42 log]# yum install python26

epel
| 3.7 kB     00:00
epel/primary_db
| 3.8 MB     00:00
ius
| 2.2 kB     00:00
ius/primary_db
| 122 kB     00:00
Setting up Install Process
Resolving Dependencies
--> Running transaction check
---> Package python26.x86_64 0:2.6.8-2.el5 set to be updated
--> Processing Dependency: libpython2.6.so.1.0()(64bit) for package: python26
--> Processing Dependency: libffi.so.5()(64bit) for package: python26
--> Running transaction check
---> Package libffi.x86_64 0:3.0.5-1.el5 set to be updated
---> Package python26-libs.x86_64 0:2.6.8-2.el5 set to be updated
--> Finished Dependency Resolution

Dependencies Resolved

==========================================================================================
==========================================================================================
==================
 Package                                                 Arch
Version                                                  Repository
Size
==========================================================================================
==========================================================================================
==================
Installing:
 python26                                                x86_64
2.6.8-2.el5                                              epel
6.5 M
Installing for dependencies:
 libffi                                                  x86_64
3.0.5-1.el5                                              epel
24 k
```

```
 python26-libs                                        x86_64
2.6.8-2.el5                                           epel
695 k

Transaction Summary
=========================================================================================
=========================================================================================
==================
Install       3 Package(s)
Upgrade       0 Package(s)

Total download size: 7.2 M
Is this ok [y/N]:
Is this ok [y/N]: y
Downloading Packages:
(1/3): libffi-3.0.5-1.el5.x86_64.rpm
|  24 kB     00:00
(2/3): python26-libs-2.6.8-2.el5.x86_64.rpm
| 695 kB     00:00
(3/3): python26-2.6.8-2.el5.x86_64.rpm
| 6.5 MB     00:00
-----------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------
------------------
Total
3.4 MB/s | 7.2 MB     00:02
warning: rpmts_HdrFromFdno: Header V3 DSA signature: NOKEY, key ID 217521f6
epel/gpgkey
| 1.7 kB     00:00
Importing GPG key 0x217521F6 "Fedora EPEL <epel@fedoraproject.org>" from /etc/pki/rpm-
gpg/RPM-GPG-KEY-EPEL
Is this ok [y/N]:
Is this ok [y/N]: y
Running rpm_check_debug
Running Transaction Test
Finished Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing     : libffi
1/3
  Installing     : python26
2/3
  Installing     : python26-libs
3/3

Installed:
  python26.x86_64 0:2.6.8-2.el5

Dependency Installed:
  libffi.x86_64 0:3.0.5-1.el5
python26-libs.x86_64 0:2.6.8-2.el5

Complete!
```

```
[root@ibmx3650-svl42 log]#
```

3. Verify after python26 is installed.

```
[root@ibmx3650-svl42 bin]# rpm -ql python26 | grep '/usr/bin'
/usr/bin/pydoc26
/usr/bin/python2.6
/usr/bin/python26
```

8.3 Configuring NetApp Scheduler Plug-In

The NetApp scheduler plug-in consists of `schmod_netapp.so`, `ntapplugin.conf`, `ontapmon.py`, and `config.ini` files. When uncompressed in the `LSF_TOP` location on the LSF master node, it creates its own folder called `netapp_lsf_plugin_v2.0`. After the external scheduler plug-in – `schmod_netapp.so` is compiled, it creates a bunch of object files `(.o)`. The `misc` directory contains the important files such as `config.ini`, `ntapplugin.conf`, `ontapmon.py`, and `ntapplugin.conf`. Later the `ntapplugin.conf` is copied into the `$LSF_ENVDIR` location.

1. The plug-in must be built on the LSF master node. Get to the `netapp_lsf_plugin_v2.0` folder in the `LSF_TOP` that has all the files that are required to build the plug-in. The include files in this paper reside in location `/mnt/lsf-root/share/lsf/8.0/include/lsf`.

```
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# ls
libntap.c  Makefile  misc  README.txt  sysntap.c  sysntap.h  tools
```

2. Edit the `CFLAG` variable in the `Makefile` to point to the location where all the LSF `include` files reside.

```
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# vi Makefile

# make make make
#
CC = gcc -ggdb
CFLAGS = -I/mnt/lsf-root/share/lsf/8.0/include/lsf -fPIC -I./tools
LDFLAGS = -shared
LIBS = -lm
PLUGIN = schmod_netapp.so
TOOLS = tools/libtools.a
LIBEXPAT = tools/expat-2.0.1/.libs/libexpat.a


....................................
```

3. Run `make` to build the plug-in.

```
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# make
gcc -ggdb -I/mnt/lsf-root/share/lsf/8.0/include/lsf -fPIC -I./tools -c libntap.c
gcc -ggdb -I/mnt/lsf-root/share/lsf/8.0/include/lsf -fPIC -I./tools -c sysntap.c
sysntap.h
cd tools; make
make[1]: Entering directory `/mnt/lsf-root/share/lsf/netapp_lsf_plugin_v2.0/tools'
gcc -g -fPIC -c conf.c conf.h tools.h
gcc -g -fPIC -c tab.c tab.h tools.h
gcc -g -fPIC -c msg.c msg.h tools.h
```

```
ar r libtools.a conf.o tab.o msg.o
ar: creating libtools.a
make[1]: Leaving directory `/mnt/lsf-root/share/lsf/netapp_lsf_plugin_v2.0/tools'
cd tools/expat-2.0.1; ./configure --with-pic; make
checking build system type... x86_64-unknown-linux-gnu
checking host system type... x86_64-unknown-linux-gnu
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ANSI C... none needed
checking for a sed that does not truncate output... /bin/sed
checking for egrep... grep -E
checking for ld used by gcc... /usr/bin/ld
checking if the linker (/usr/bin/ld) is GNU ld... yes
checking for /usr/bin/ld option to reload object files... -r
checking for BSD-compatible nm... /usr/bin/nm -B
checking whether ln -s works... yes
checking how to recognise dependent libraries... pass_all
checking how to run the C preprocessor... gcc -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
checking for strings.h... yes
checking for inttypes.h... yes
checking for stdint.h... yes
checking for unistd.h... yes
checking dlfcn.h usability... yes
checking dlfcn.h presence... yes
checking for dlfcn.h... yes
checking for g++... g++
checking whether we are using the GNU C++ compiler... yes
checking whether g++ accepts -g... yes
checking how to run the C++ preprocessor... g++ -E
checking for g77... g77
checking whether we are using the GNU Fortran 77 compiler... yes
checking whether g77 accepts -g... yes
checking the maximum length of command line arguments... 32768
checking command to parse /usr/bin/nm -B output from gcc object... ok
checking for objdir... .libs
checking for ar... ar
checking for ranlib... ranlib
checking for strip... strip
checking if gcc supports -fno-rtti -fno-exceptions... no
checking for gcc option to produce PIC... -fPIC
checking if gcc PIC flag -fPIC works... yes
```

```
checking if gcc static flag -static works... yes
checking if gcc supports -c -o file.o... yes
checking whether the gcc linker (/usr/bin/ld -m elf_x86_64) supports shared
libraries... yes
checking whether -lc should be explicitly linked in... no
checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs... immediate
checking whether stripping libraries is possible... yes
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... yes
configure: creating libtool
appending configuration tag "CXX" to libtool
checking for ld used by g++... /usr/bin/ld -m elf_x86_64
checking if the linker (/usr/bin/ld -m elf_x86_64) is GNU ld... yes
checking whether the g++ linker (/usr/bin/ld -m elf_x86_64) supports shared
libraries... yes
checking for g++ option to produce PIC... -fPIC
checking if g++ PIC flag -fPIC works... yes
checking if g++ static flag -static works... yes
checking if g++ supports -c -o file.o... yes
checking whether the g++ linker (/usr/bin/ld -m elf_x86_64) supports shared
libraries... yes
checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs... immediate
appending configuration tag "F77" to libtool
checking if libtool supports shared libraries... yes
checking whether to build shared libraries... yes
checking whether to build static libraries... yes
checking for g77 option to produce PIC... -fPIC
checking if g77 PIC flag -fPIC works... yes
checking if g77 static flag -static works... yes
checking if g77 supports -c -o file.o... yes
checking whether the g77 linker (/usr/bin/ld -m elf_x86_64) supports shared
libraries... yes
checking dynamic linker characteristics... GNU/Linux ld.so
checking how to hardcode library paths into programs... immediate
checking for gcc... (cached) gcc
checking whether we are using the GNU C compiler... (cached) yes
checking whether gcc accepts -g... (cached) yes
checking for gcc option to accept ANSI C... (cached) none needed
checking for a BSD-compatible install... /usr/bin/install -c
checking whether gcc accepts -fexceptions... yes
checking for ANSI C header files... (cached) yes
checking whether byte ordering is bigendian... no
checking for an ANSI C-conforming const... yes
checking for size_t... yes
checking for memmove... yes
checking for bcopy... yes
checking fcntl.h usability... yes
checking fcntl.h presence... yes
checking for fcntl.h... yes
checking for unistd.h... (cached) yes
```

```
checking for off_t... yes
checking for stdlib.h... (cached) yes
checking for unistd.h... (cached) yes
checking for getpagesize... yes
checking for working mmap... yes
checking for an ANSI C99-conforming __func__... yes
configure: creating ./config.status
config.status: creating Makefile
config.status: creating expat_config.h
config.status: expat_config.h is unchanged
make[1]: Entering directory `/mnt/lsf-
root/share/lsf/netapp_lsf_plugin_v2.0/tools/expat-2.0.1'
/bin/sh ./libtool --silent --mode=compile gcc -I./lib -I. -g -O2 -Wall -Wmissing-
prototypes -Wstrict-prototypes -fexceptions  -DHAVE_EXPAT_CONFIG_H -o
lib/xmlparse.lo -c lib/xmlparse.c
/bin/sh ./libtool --silent --mode=compile gcc -I./lib -I. -g -O2 -Wall -Wmissing-
prototypes -Wstrict-prototypes -fexceptions  -DHAVE_EXPAT_CONFIG_H -o lib/xmltok.lo
-c lib/xmltok.c
/bin/sh ./libtool --silent --mode=compile gcc -I./lib -I. -g -O2 -Wall -Wmissing-
prototypes -Wstrict-prototypes -fexceptions  -DHAVE_EXPAT_CONFIG_H -o
lib/xmlrole.lo -c lib/xmlrole.c
/bin/sh ./libtool --silent --mode=link gcc -I./lib -I. -g -O2 -Wall -Wmissing-
prototypes -Wstrict-prototypes -fexceptions  -DHAVE_EXPAT_CONFIG_H -no-undefined -
version-info 6:2:5 -rpath /usr/loca   l/lib  -o libexpat.la lib/xmlparse.lo
lib/xmltok.lo lib/xmlrole.lo
gcc -I./lib -I. -g -O2 -Wall -Wmissing-prototypes -Wstrict-prototypes -fexceptions
-DHAVE_EXPAT_CONFIG_H -o xmlwf/xmlwf.o -c xmlwf/xmlwf.c
gcc -I./lib -I. -g -O2 -Wall -Wmissing-prototypes -Wstrict-prototypes -fexceptions
-DHAVE_EXPAT_CONFIG_H -o xmlwf/xmlfile.o -c xmlwf/xmlfile.c
gcc -I./lib -I. -g -O2 -Wall -Wmissing-prototypes -Wstrict-prototypes -fexceptions
-DHAVE_EXPAT_CONFIG_H -o xmlwf/codepage.o -c xmlwf/codepage.c
gcc -I./lib -I. -g -O2 -Wall -Wmissing-prototypes -Wstrict-prototypes -fexceptions
-DHAVE_EXPAT_CONFIG_H -o xmlwf/unixfilemap.o -c xmlwf/unixfilemap.c
/bin/sh ./libtool --silent --mode=link gcc -I./lib -I. -g -O2 -Wall -Wmissing-
prototypes -Wstrict-prototypes -fexceptions  -DHAVE_EXPAT_CONFIG_H  -o xmlwf/xmlwf
xmlwf/xmlwf.o xmlwf/xmlfile.o xmlw   f/codepage.o xmlwf/unixfilemap.o libexpat.la
make[1]: Leaving directory `/mnt/lsf-
root/share/lsf/netapp_lsf_plugin_v2.0/tools/expat-2.0.1'
gcc -ggdb -shared  -o schmod_netapp.so libntap.o sysntap.o tools/libtools.a
tools/expat-2.0.1/.libs/libexpat.a -lm
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]#
```

4. Now a new `schmod_netapp.so` is created in the plug-in location where the make was run. Copy the `schmod_netapp.so` file into the `$LSF_LIBDIR`.

```
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# ls
libntap.c  libntap.o  Makefile  misc  README.txt  schmod_netapp.so  sysntap.c
sysntap.h  sysntap.h.gch  sysntap.o  tools

 [root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]#

[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# cp schmod_netapp.so $LSF_LIBDIR

[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# ls $LSF_LIBDIR
```

```
cal_jobweight.so     jsdl.xsd         libesc.so               libpm.a
schmod_aps.so       schmod_dist.so      schmod_netapp.so      schmod_rms.so
daemons_old          lib2vemkd.so     libevent.so             libptmalloc3.so
schmod_bluegene.so  schmod_fairshare.so  schmod_parallel.so    schmod_slurm.so
esd_ego_default.so   libbat.a         libfairshareadjust.so  librbac.so
schmod_cpuset.so    schmod_fcfs.so      schmod_preemption.so  sec_ego_default.so
eventplugin_snmp.so  libbat.jsdl.a    liblsbstream.so        libvem.so
schmod_crayx1.so    schmod_jobweight.so  schmod_pset.so        sec_ego_ext_co.so
jsdl-lsf.xsd         libbat.so        liblsf.a               rbac_ego_default.so
schmod_crayxt3.so   schmod_limit.so     schmod_ps.so
jsdl-posix.xsd       libegostream.so  liblsf.so              schmod_advrsv.so
schmod_default.so   schmod_mc.so        schmod_reserve.so
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]#
```

5. The LSF environment has to be configured with the new module so that it can register with the LSF scheduler as an external plug-in.

```
[root@ibmx3650-svl50 DIRLOC]# vi $LSF_ENVDIR/lsf.conf
LSB_SBD_PORT=6882

# WARNING: Please do not delete/modify next line!!
LSF_LINK_PATH=n

# LSF_MACHDEP and LSF_INDEP are reserved to maintain
# backward compatibility with legacy lsfsetup.
# They are not used in the new lsfinstall.
LSF_INDEP=/mnt/lsf-root/share/lsf
LSF_MACHDEP=/mnt/lsf-root/share/lsf/8.0

LSF_TOP=/mnt/lsf-root/share/lsf
LSF_VERSION=8.0
LSF_ENABLE_EGO=N
# LSF_EGO_ENVDIR=/mnt/lsf-root/share/lsf/conf/ego/lsf-cluster1/kernel
EGO_WORKDIR=/mnt/lsf-root/share/lsf/work/lsf-cluster1/ego
LSF_LIVE_CONFDIR=/mnt/lsf-root/share/lsf/work/lsf-cluster1/live_confdir
LSF_MASTER_LIST="ibmx3650-svl50"
LSF_EGO_DAEMON_CONTROL=N
LSF_LICENSE_FILE=/mnt/lsf-root/share/lsf/conf/license.dat
LSF_RSH=ssh
LSF_ENABLE_EXTSCHEDULER=y
```

6. Include the `schmod_netapp.so` module in the `lsb.modules`.

```
[root@ibmx3650-svl50 netapp_lsf_plugin_v2.0]# vi /mnt/lsf-
root/share/lsf/conf/lsbatch/lsf-cluster1/configdir/lsb.modules
# $Id: lsb.modules,v 1.9 2007/02/22 19:54:59 lguo Exp $

# Define plugins for Scheduler and Resource Broker.
# SCH_PLUGIN coloum specifies the share module name for Scheduler, while
# RB_PLUGIN specifies the share module name for Resource Broker
# A Scheduler plugin can have one, multiple, or none RB plugins
# corresponding to it.
# SCH_DISABLE_PHASES specifies which phases of that scheduler plugin
```

```
# should be disabled, i.e., inactivated. A scheduler plugin has 4 phases:
# pre processing, match/limit, order/alloc, post processing. Scheduler
# won't invokes disabled phases over jobs
# Note all share modules should be put under LSF_LIBDIR

Begin PluginModule
SCH_PLUGIN                      RB_PLUGIN                    SCH_DISABLE_PHASES
schmod_default                  ()                              ()
schmod_fcfs                     ()                              ()
schmod_fairshare                ()                              ()
schmod_limit                    ()                              ()
schmod_parallel                 ()                              ()
schmod_reserve                  ()                              ()
schmod_mc                       ()                              ()
schmod_preemption               ()                              ()
schmod_advrsv                   ()                              ()
schmod_ps                       ()                              ()
schmod_netapp                   ()                              ()
End PluginModule
```

7. Edit the `ntapplugin.conf` file for the `schmod_netapp.so` to read the following configuration information. The highlighted parameters `ntapplugin.conf` file needs to be changed:
    a. Controllers that it needs to query the DFM server.
    b. Volumes that need to be monitored.
    c. Update the Counter directory location. In this location the `ontapmon.py` script is going to write the `<NetApp_controller_hostname>.XML` file and error logs. In this paper the `fas6280c-svl05.xml` file is located in `/var/log/plugin/DIRLOC`.
    d. Verify the threshold values set for the global and per volume parameters.
    e. Set up the working directory location. The ntapplugin.log file is written in this location. In this paper the working directory is referred to `/var/log/plugin/netapp-log`. After the NetApp monitoring script starts, it generates an `ntapplugin.log` file in the working directory `/var/log/plugin/netapp-log`.

```
[root@ibmx3650-svl50 conf]# vi ntapplugin.conf
#
# $Id: filerplugin.conf,v 1.1 2010/12/30 22:46:23 david Exp $
#
# /etc/filesystemtags
#
# Interface defining the filers and file systems to the scheduling plugin
# this file has to be managed by the site system administrator.
# This file has two sections:
#
# First section lists the available filers filesystems
#
Begin ExportNames
lsf_storage     fas6280c-svl05:/vol/USERVOL_16d_rg,fas6280c-
svl05:/vol/eda_16d_rg,fas6280c-svl05:/vol/VCS_16d_rg,fas6280c-svl05:/vol/lsf_16d_rg
# test2 fas6280c-svl12:/vol/nfsDS
End ExportNames

#
# The second section lists the GLOBAL filer utilization thresholds
```

```
# and file system space thresholds.
#

Begin PluginPolicy
Max_DiskBusy     =       50
Max_NEDomain     =       75
Max_AvgVolLatency =      20
Min_AvailFiles   =       1000
Min_AvailSize    =       1000
End PluginPolicy

#
# Section where one can define Volume and Filer specific parameters
# and policies.  IF Filer or volume are not specified then, global
# thresholds will be used.
#
Begin FilerPolicy
fas6280c-svl05:/vol/USERVOL_16d_rg      Min_AvailFiles = 1000
fas6280c-svl05:/vol/USERVOL_16d_rg      Min_AvailSize = 1000
fas6280c-svl05:/vol/USERVOL_16d_rg      Max_DiskBusy =  50
fas6280c-svl05:/vol/USERVOL_16d_rg      Max_AvgVolLatency = 15
End FilerPolicy

#
# Parameter section controlling plugin
# behaviour.
Begin Parameters
Debug yes
Work_Dir /var/log/plugin/netapp-log
Counter_Dir /var/log/plugin/DIRLOC
XMLReread 60
DryRunMode no
End Parameters
```

8. The `config.ini` file needs to be configured in order for the NetApp monitoring script (`ontapmon.py`) to read the login credentials and the frequency in which it is going to poll the DFM server. The location of the `NMSDK` should also be provided.

```
[root@ibmx3650-svl50 misc]# vi config.ini
[env_params]
NMDKDIR = /mnt/lsf-root/share/lsf/netapp-manageability-sdk-5.0R1
[dfm_param]
HOST = 172.17.44.231
USER = EDA\administrator
PASSWD = Netapp12345
[mon_param]
INTERVAL = 60
DIRLOC = /var/log/plugin/DIRLOC
NTHREADS = 4
REFRESH = 5
```

The NetApp monitoring script queries the DFM server every 60 seconds as per the default `interval` value set in the `config.ini` file. The `interval` value can be set as low as 30 seconds to query data more frequently. For more details on the other parameters, read the `README` file provided with the plug-in.

9. Whenever the `ntapplugin.conf` and `config.ini` files are changed, the following commands must be issued for the NetApp scheduler plug-in (`schmod_netapp.so`) to register with the LSF scheduler with the updated values.

```
[root@ibmx3650-svl50 misc]# lsadmin limrestart

Checking configuration files ...
No errors found.

Restart LIM on <ibmx3650-svl50.iop.eng.netapp.com> ...... done


[root@ibmx3650-svl50 misc]# badmin mbdrestart

Checking configuration files ...

No errors found.

MBD restart initiated
```

10. Start the Data ONTAP monitoring script (`ontapmon.py`) with the following command and verify it. Run the script in its current location with the `python` command along with the `config.ini` file.

```
[root@ibmx3650-svl50 misc]# python ontapmon.py config.ini &
[2] 30761

[root@ibmx3650-svl50 misc]# ps -ef|grep python
root      3991     1  0 Feb17 ?        00:00:00 /usr/bin/python ./hpssd.py
root      4640     1  0 Feb17 ?        00:00:07 /usr/bin/python -tt /usr/sbin/yum-
updatesd
root     29127 29125  0 16:00 ?        00:00:01 /usr/bin/python26 /mnt/lsf-
root/share/lsf/8.0/linux2.6-glibc2.3-x86_64/etc/elim.netapp_compute
root     30761 32729  0 17:35 pts/3    00:00:00 python ontapmon.py config.ini
root     30773 32729  0  17:36 pts/3    00:00:00 grep python
```

11. The plug-in is activated for submitting jobs to the LSF scheduler by using the `bsub –extsched`

```
bsub –extsched “filer[lsf_storage1,lsf_storage2…]” ….
```

Where `lsf_storage1` is the export name specified in the `ntapplugin.conf` file.

12. The `fas6280c-svl05.xml` and `ontapmon_error.log` file is created in the `/var/log/plugin/DIRLOC/` location.

```
[root@ibmx3650-svl50 DIRLOC]# ls -l
total 4952
-rw-r--r-- 1 root root   3319 Mar 19 02:46 fas6280c-svl05.xml
-rw-r--r-- 1 root root   1289 Mar 19 02:46 fas6280c-svl06.xml
-rw-r--r-- 1 root root  88138 Mar 19 02:48 ontapmon_error.log
```

13. The `ntapplugin.log` file will show all the threshold values that are set globally on the NetApp storage and also values set on each volume. This will also list the schmod_netapp.so the plug-in is configured correctly. This information will be logged in the `ntapplugin.log` file if the debug mode is set in the `ntapplugin.conf` file. The debug mode is enabled by default.

```
[root@ibmx3650-svl50 netapp-log]# tail -f ntapplugin.log
Mar 20 20:10:20:804066 9881 parse_policies():  Max_NEDomain 1000.000 configured
Mar 20 20:10:20:804096 9881 parse_filerpolicies():  Max_NEDomain 1000.000
configured
Mar 20 20:10:20:804108 9881 parse_filerpolicies():  Max_NEDomain 1000.000
configured
Mar 20 20:10:20:804117 9881 parse_filerpolicies():  Max_DiskBusy 50.000 configured
Mar 20 20:10:20:804126 9881 parse_filerpolicies():  Max_AvgVolLat_Busy 15.000
configured
Mar 20 20:10:20:804134 9881 filertab->size 269

Mar 20 20:10:20:804140 9881 146: data volume fas6280c-svl05:/vol/USERVOL_16d_rg
50.000000 0.000000 15.000000 1000.000000 1000.000000

Mar 20 20:10:20:804160 9881 read_conf(): schmod_ntap.so plugin configured all right
```

8.4 Job Handling by NetApp Scheduler Plug-In

When the NetApp storage is quiet or the resource utilization is below the threshold values set, jobs submitted through "`bsub`" start to run. After the resource utilization goes beyond the threshold values, the plug-in sets a `PEND` status on jobs that will target one or more of the affected volumes.

In the following example the threshold value for disk utilization was set to 30%. After the NetApp storage crossed the threshold value, the NetApp scheduler set a `PEND` status on the job ID 5173 and other jobs thereafter until the disk utilization drops under 30%.

```
Mar 28 16:23:42:587455 22880 compare_thresholds(): volume USERVOL_16d_rg: diskb
86.21800, avglatency 2.92480, availinodes 5188957.00000, availsize
266433269760.00000

Mar 28 16:23:42:587465 22880 allocate(): filer load not ok jobID 5173 mount
lsf_storage SCH_MOD_DECISION_PENDJOB
```

If all volumes in the groups targeted by the job are free of performance problems, the plug-in will set a `DISPATCH` status on the job. The status can be verified in the `ntapplugin.log` file in the working directory as configured in the `ntapplugin.conf` file. In this paper the working directory is referred to /`var/log/plugin/netapp-log`.

The following example illustrates how the status for job ID 5173 is changed to DISPATCH after the disk utilization dropped below the threshold value of 30%.

```
Mar 28 16:25:53:400050 22880 compare_thresholds(): volume lsf_16d_rg: diskb
19.05600, avglatency 0.01726, availinodes 31122709.00000, availsize
1070302277632.00000

Mar 28 16:25:53:400060 22880 allocate(): filer load ok jobID 5173 mount
lsf_storage SCH_MOD_DECISION_DISPATCH
```

The DISPATCH and PEND status on the jobs submitted by “bsub” command from the LSF scheduler is dependent on two conditions:

- The “bsub” command checks the NetApp scheduler plug-in when the jobs are first submitted. For example, there are 100 jobs is submitted by the “bsub.” After 20 jobs are processed, the storage resources cross the threshold values. The NetApp scheduler plug-in is not going to set a PEND state on job 21 and rest of the jobs at that time. The LSF scheduler is going to continue processing all the 100 jobs till it completes.
- There must be some free slots (cores) for the LSF scheduler to submit the jobs. If all the slots are used up for a particular “bsub,” then the subsequent “bsub” job submissions will PEND by the LSF scheduler anyway until the slots are freed up. In this condition there will be no PEND state on the jobs by the NetApp scheduler plug-in. The jobs will be all set to a DISPATCH state. The jobs will start to run when the LSF scheduler finds any free slots and the NetApp scheduler still continues to find that the storage resources are below threshold values.

8.5 Configuring NetApp Compute Agent and Hot Job Detection Plug-In

As listed earlier, the NetApp compute agent and hot job detection plug-in consist of `elim.netapp_compute`, `netapp_lsf_hot_job_detector.py`, `netapp_lsf_hot_job_detector.conf`, and `netapp_lsf_hot_job_email.py` files. After all the files are uncompressed in the `LSF_TOP` directory location, perform the following to complete the configuration.

1. Include the `netapp_compute` function in the `lsb.shared` and `lsf.cluster.lsf-cluster1` files. In this paper `lsf-cluster1` is the name of the LSF cluster. The Boolean is a dummy value that does not return any parameter to the LSF scheduler.

```
[root@ibmx3650-svl50 lsf]# vi /mnt/lsf-root/share/lsf/conf/lsf.shared

define_ncpus_threads Boolean () ()      (ncpus := threads)
#  ostype     String  ()       ()          (Operating System and version)
#  lmhostid   String  ()       ()          (FLEXLM's lmhostid)
#  limversion String  ()       ()          (Version of LIM binary)
   netapp_compute       Boolean 15      () (NetApp Compute Agent)
End Resource

[[root@ibmx3650-svl50 lsf]# vi /mnt/lsf-root/share/lsf/conf/lsf.cluster.lsf-
cluster1
Begin ResourceMap
  RESOURCENAME  LOCATION
# tmp2          [default]
# nio           [all]
# console       [default]
# osname        [default]
# osver         [default]
# cpuarch       [default]
# cpuspeed      [default]
# bandwidth     [default]
  netapp_compute  [default]
End ResourceMap
```

2. Move the `elim.netapp_compute` file to `$LSF_SERVERDIR` location. Change the permissions on the `elim.netapp_compute` file.

```
[root@ibmx3650-svl50 lsf]#  mv netapp_lsf_compute_agent.py
netapp_lsf_compute_agent.conf $LSF_SERVERDIR
```

```
[root@ibmx3650-svl50 lsf]# mv /mnt/lsf-root/share/lsf/8.0/linux2.6-glibc2.3-x86_64/etc/netapp_lsf_compute_agent.py $LSF_SERVERDIR/elim.netapp_compute

[root@ibmx3650-svl50 lsf]# chmod +x $LSF_SERVERDIR/elim.netapp_compute
```

3. A compiled `SystemTap` module needs to be created for each kernel version on which the compute agent will run. Installing a complete `SystemTap` package on the master LSF node was already discussed in an earlier section. If there are multiple kernels in the compute farm, then `SystemTap` has to be completed for each of those kernels. The rest of the nodes in the compute farm should have the `SystemTap runtime` package to run the compiled module.

```
[root@ibmx3650-svl50 lsf]# stap -r 2.6.18-308.24.1.el5 netapp_nfsmon.stp -m netapp_nfsmon_2_6_18_308_24_1_el5
```

   This will generate a `.ko` file - netapp`_nfsmon_2_6_18_308_24_1_el5.ko`. This .ko file is the compiled `SystemTap` that would run on all the nodes. If there are multiple kernels, there has to be a separate .ko file for each kernel.

4. The `netapp_lsf_compute_agent.conf` file has to be modified to point to the `.ko` file that was compiled earlier and also to the location where each of the jobs would create a TXT file in the `job_report` directory. It is recommended to create the `job_report` directory on a shared location on the NetApp storage. In this paper all the TXT files are created in the `/mnt/lsf-root/share/lsf/job_report` directory.

```
[root@ibmx3650-svl50 etc]# vi netapp_lsf_compute_agent.conf
[MAIN]

; The directory containing the compiled SystemTap module (.ko file) for
; the kernel version.
systemtap_modules_directory = /mnt/lsf-root/share/lsf

; Output directory for the LSF job report text files.
job_report_output_directory = /mnt/lsf-root/share/lsf/job_reports

; How often (in seconds) LSF job performance data should be written to
; text files. Between these output intervals, the performance data is
; aggregated into periods.
performance_output_write_interval = 30

; Period of inactivity (in seconds) after which, if activity occurs,
; the LSF job number for the PID should be updated.
pid_to_lsf_job_number_expiration_time = 60

; How often (in seconds) to update the LSF cluster name of which
; the compute node is a member.
cluster_name_fetch_interval = 600
```

5. Restart the LSF LIMs to cause the compute agent ELIM script to be run on every compute host.

```
[root@ibmx3650-svl50 etc]# lsadmin reconfig

Checking configuration files ...
```

```
No errors found.

Restart only the master candidate hosts? [y/n] n
Do you really want to restart LIMs on all hosts? [y/n] y
Restart LIM on <ibmx3650-svl50.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl42.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl43.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl44.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl45.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl46.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl47.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl49.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl51.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl52.iop.eng.netapp.com> ...... done
Restart LIM on <ibmx3650-svl53.iop.eng.netapp.com> ...... done
```

6. Check if the compute agent is running fine from the `/var/log/netapp_lsf_compute_agent.log` file. This log file is generated on every compute node in the farm in the `/var/log` location. If all the configuration is completed successfully without any errors, each node will have a corresponding log file in the /var/log location that will list all the volumes that have been mounted (static or automount) on that node.

```
2013-03-20 20:23:32,738 DEBUG Retrieved device number map: {'0:20':
'172.31.22.105:/vol/VCS_28d_rg', '0:28': '172.31.22.105:/vol/lsf_16d_rg', '0:24':
'172.31.22.106:/vol/USERVOL_28d_rg', '0:26': '172.31.22.105:/vol/eda_28d_rg',
'0:27': '172.31.22.105:/vol/sge_28d_rg'}
```

Another way to check of the `SystemTap` is functional on all nodes is by verifying if `stap` process is running on each node. If `stap` is not running on any node, then the jobs running on that node will not generate any TXT file in the `/mnt/lsf-root/share/lsf/job_reports` location. Run the `lsadmin reconfig` on the master LSF node to restart the LIMs on each node that would start the `stap`.

```
[root@ibmx3650-svl43 log]# ps -ef|grep stap
root      9209  9195  0 20:14 ?        00:00:00 /usr/libexec/systemtap/stapio
/mnt/lsf-root/share/lsf/netapp_nfsmon_2_6_18_308_24_1_el5.ko
root      9553  9454  0 20:29 pts/1    00:00:00 grep stap
```

If the `stap` still does not start, checking the `/var/log/ netapp_lsf_compute_agent.log` would be the best place to start looking for any errors.

7. Edit the `netapp_lsf_hot_job_detection.conf` file to add the different storage thresholds to identify when the jobs will be identified as hot or runaway jobs. These threshold values can be the same values that are set on the `ntapplugin.conf` file in the NetApp scheduler plug-in. The `netapp_lsf_hot_job_detection.py` script when executed runs in the background.

```
[root@ibmx3650-svl50 lsf]# vi netapp_lsf_hot_job_detector.conf
[MAIN]

; The directory containing the XML files outputted by the NetApp ontapmon.py
; performance monitoring script.
ontap_xml_data_directory = /var/log/plugin/DIRLOC
```

```
; The directory path containing the LSF job report text files outputted by
; the NetApp LSF compute agent ELIM script.
lsf_job_report_directory = /mnt/lsf-root/share/lsf/job_reports

.............

If both a controller-level threshold and a volume-level threshold
; exist, the volume-level threshold takes priority. For example, in
; the sample thresholds above, volNFS on controller fas6280c-svl will
; be tested against a Max_AvgVolLatency value of 100, since that is the
; more specific threshold. Other volumes on that controller (which
; don't have a specific volume-level threshold set) will be tested against
; the controller-level Max_AvgVolLatency value of 50.



fas6280c-svl05:/vol/USERVOL_16d_rg      Min_AvailFiles = 1000
fas6280c-svl05:/vol/USERVOL_16d_rg      Min_AvailSize = 1000
fas6280c-svl05:/vol/USERVOL_16d_rg      Max_DiskBusy =  50
fas6280c-svl05:/vol/USERVOL_16d_rg      Max_AvgVolLatency = 15
```

8. The e-mail notification is configured in the `netapp_lsf_hot_job_email.py` to notify the storage administrator of any runaway jobs after the threshold values listed in the `netapp_lsf_hot_job_detection.conf` file are met.

```
[root@ibmx3650-svl50 lsf]# vi netapp_lsf_hot_job_email.py
from email.mime.text import MIMEText

toAddresses = ['bikash@netapp.com']
fromAddress = 'lsf-reports@netapp.com'
smtpServer = 'smtp.corp.netapp.com'


 server = smtplib.SMTP(smtpServer)
        server.sendmail(fromAddress, toAddresses, msg.as_string())
        server.quit()
```

9. Change the permissions and start the netapp_lsf_hot_job_dection.py script. This script will call the netapp_lsf_hot_job_email.py script when any jobs cross the threshold values.

```
[root@ibmx3650-svl50 lsf]# chmod +x netapp_lsf_hot_job_detector.py
[root@ibmx3650-svl50 lsf]#
[root@ibmx3650-svl50 lsf]# ./netapp_lsf_hot_job_detector.py &
[2] 25913
```

10. Check if the hot_job_detection script is running in the background and a new log file `netapp_lsf_hot_job_detector.log` is created in the `/var/log` location of the master LSF node.

```
[root@ibmx3650-svl50 lsf]# ps -ef|grep python
root      3991     1  0 Feb17 ?        00:00:00 /usr/bin/python ./hpssd.py
root      4640     1  0 Feb17 ?        00:00:40 /usr/bin/python -tt /usr/sbin/yum-
updatesd
```

```
root     25201 32729  0 12:21 pts/3    00:00:01 python ontapmon.py config.ini
root     25693 25691  0 12:43 ?        00:00:00 /usr/bin/python26 /mnt/lsf-
root/share/lsf/8.0/linux2.6-glibc2.3-x86_64/etc/elim.netapp_compute
root     25913 32729  0 12:48 pts/3    00:00:00 /usr/bin/python26
./netapp_lsf_hot_job_detector.py
root     25919  4640 48 12:48 ?        00:00:06 /usr/bin/python -tt
/usr/libexec/yum-updatesd-helper --check --dbus
root     25933 32729  0 12:48 pts/3    00:00:00 grep python
```

8.6 Job Handling by NetApp Compute Agent and Hot Job Detection Plug-In

The `netapp_hot_job_detector.py` script reads the XML output from the NetApp monitoring script xml to gather information on the NetApp storage resources. It compares these performance values against the thresholds set in the `netapp_hot_job_detector.conf` file. The volumes listed in the XML data for each controller are checked in the same order as listed in the XML file.

For example, there are four volumes in an aggregate. A maximum disk utilization threshold of 30% is set on one of the four volumes in the configuration file, and a controller-level threshold of 50% is also set.

1. The `netapp_hot_job_detector.py` script reads the XML file for the controller and checks the first volume, which has a disk utilization value of 40%, against the configured thresholds.
2. A volume-level threshold is not set on this volume, so the controller-level threshold of 50% applies. No performance problem is detected.
3. The process repeats, checking each of the four volumes, until the script checks the volume with the volume-level threshold of 30%.
4. The script detects that the volume-level maximum disk utilization threshold of 30% has been exceeded, and an e-mail notification is triggered.

```
-----Original Message-----
From: NetApp.LSF.Report@netapp.com [mailto:NetApp.LSF.Report@netapp.com]
Sent: Friday, March 29, 2013 8:16 AM
To: Roy Choudhury, Bikash
Subject: NetApp LSF Hot Job Report

The NetApp LSF Hot Job Detection script has discovered the following performance
problems based on the configured thresholds. The LSF jobs performing the most
operations on the impacted target(s) are listed below.

**********
Controller: fas6280c-svl05

       -Aggregate aggr3 has a disk that has exceeded the threshold for acceptable
maximum disk busy. Threshold: 50.00, Value: 74.67.

       Top jobs operating on aggregate fas6280c-svl05:aggr3:
              -LSF job number 5383 on cluster lsf-cluster1.txt has performed 81151
recent operations on target (RD = 81151, WR = 0).
              -LSF job number 5386 on cluster lsf-cluster1.txt has performed 79639
recent operations on target (RD = 79639, WR = 0).
              -LSF job number 5389 on cluster lsf-cluster1.txt has performed 78565
recent operations on target (RD = 78565, WR = 0).
```

After the hot job IDs are identified, an LSF administrator can query LSF for the job details to find the user who submitted the job and the host on which the job is running. Necessary action may be taken on the hot or runaway job by the administrator.

```
[lsfadmin@ibmx3650-svl50 farm_cpu_test]$ bjobs -l 5389
```

```
Job <5389>, User <lsfadmin>, Project <default>, Status <RUN>, Queue <normal>, E
                     xtsched <filer[lsf_storage]>, Command <./readMPDrand.pl re
                     adMPDrand.log.15>
Fri Mar 29 08:13:02: Submitted from host <ibmx3650-svl50.iop.eng.netapp.com>, C
                     WD </mnt/user/OpenSPARCT1/VCS-Cloud_free_trial_demo/OpenSp
                     arc-T1/model_dir/farm_cpu_test>;
Fri Mar 29 08:13:07: Started on <ibmx3650-svl44.iop.eng.netapp.com>, Execution
                     Home </home/lsfadmin>, Execution CWD </mnt/user/OpenSPARCT
                     1/VCS-Cloud_free_trial_demo/OpenSparc-T1/model_dir/farm_cp
                     u_test>;
Fri Mar 29 08:15:53: Resource usage collected.
                     The CPU time used is 114 seconds.
                     MEM: 4 Mbytes;  SWAP: 156 Mbytes;  NTHREAD: 4
                     PGID: 10403;  PIDs: 10403 10406 10411
 SCHEDULING PARAMETERS:
           r15s   r1m  r15m   ut      pg    io   ls    it    tmp    swp    mem
 loadSched   -     -     -     -       -     -    -     -     -      -      -
 loadStop    -     -     -     -       -     -    -     -     -      -      -

 EXTERNAL MESSAGES:
 MSG_ID FROM       POST_TIME      MESSAGE                            ATTACHMENT
 0          -             -                         -                     -
 1      lsfadmin   Mar 29 08:13   filer[lsf_storage]                      N
[lsfadmin@ibmx3650-svl50 farm_cpu_test]$
```

9 Conclusion

The NetApp LSF plug-in is a step in the right direction for all the customers who use LSF scheduler and NetApp storage in EDA manufacturing environments. This plug-in makes the LSF scheduler more “storage-aware” of the resource utilization. The LSF scheduler can now make more informed decisions to pend or dispatch as the number of jobs submitted in the compute farm keeps on growing. The plug-in make a positive impact in any EDA manufacturing environment that uses LSF scheduler by:

- Reducing the number of job failures
- Avoiding resubmission of jobs
- Improving overall efficiency and ROI

The NetApp LSF plug-in will improve the overall productivity in a Synopsys VCS and similar engineering compute farm environments that are mostly running on Data ONTAP 7-Mode at this time.

Appendix

The contents of the fas6280c-svl05.xml file. The highlighted boxes in green show the volume information that is gathered from the DFM server.


```
[root@ibmx3650-svl50 DIRLOC]# cat fas6280c-svl05.xml
<?xml version='1.0' encoding='UTF-8'?>
<performance>
  <filer>fas6280c-svl05</filer>
  <lastUpdated>Tue Mar 19 02:46:06 2013</lastUpdated>
  <ipaddresses>
    <ipaddress>172.17.40.207</ipaddress>
    <ipaddress>172.17.44.48</ipaddress>
    <ipaddress>127.0.0.1</ipaddress>
    <ipaddress>127.0.20.1</ipaddress>
    <ipaddress>172.31.22.106</ipaddress>
  </ipaddresses>
  <aggregates>
    <aggr>
      <name>aggr_ssd</name>
      <maxdiskb>0.02800</maxdiskb>
      <volumes>
        <volume>
          <name>sge_28d_rg</name>
          <avglatency>0.00600</avglatency>
          <availsize>13744578560</availsize>
          <availinodes>31128277</availinodes>
        </volume>
        <volume>
          <name>eda_28d_rg</name>
          <avglatency>0.00500</avglatency>
          <availsize>29065601024</availsize>
          <availinodes>31117306</availinodes>
        </volume>
        <volume>
          <name>USERVOL_28d_rg</name>
          <avglatency>0.01265</avglatency>
          <availsize>272489267200</availsize>
          <availinodes>5188708</availinodes>
        </volume>
        <volume>
          <name>VCS_28d_rg</name>
          <avglatency>0.00525</avglatency>
          <availsize>23823831040</availsize>
          <availinodes>31098968</availinodes>
        </volume>
      </volumes>
    </aggr>
    <aggr>
      <name>aggr0</name>
      <maxdiskb>2.12000</maxdiskb>
      <volumes>
        <volume>
          <name>vol0</name>
```

```
        <avglatency>0.01377</avglatency>
        <availsize>710630166528</availsize>
        <availinodes>21977788</availinodes>
      </volume>
    </volumes>
  </aggr>
  <aggr>
    <name>aggr3</name>
    <maxdiskb>0.71500</maxdiskb>
    <volumes>
      <volume>
        <name>eda_16d_rg</name>
        <avglatency>0.00500</avglatency>
        <availsize>29065625600</availsize>
        <availinodes>31117306</availinodes>
      </volume>
      <volume>
        <name>sge_16d_rg</name>
        <avglatency>0.00500</avglatency>
        <availsize>13736542208</availsize>
        <availinodes>31128277</availinodes>
      </volume>
      <volume>
        <name>VCS_16d_rg</name>
        <avglatency>0.00500</avglatency>
        <availsize>23860809728</availsize>
        <availinodes>31098968</availinodes>
      </volume>
      <volume>
        <name>USERVOL_16d_rg</name>
        <avglatency>0.00857</avglatency>
        <availsize>266466344960</availsize>
        <availinodes>5188957</availinodes>
      </volume>
      <volume>
        <name>lsf_16d_rg</name>
        <avglatency>0.00500</avglatency>
        <availsize>1070308200448</availsize>
        <availinodes>31122856</availinodes>
      </volume>
    </volumes>
  </aggr>
</aggregates>
<domains>
  <domain>
    <name>raid</name>
    <value>5.43100</value>
  </domain>
  <domain>
    <name>target</name>
    <value>0.00800</value>
  </domain>
  <domain>
```

```
      <name>kahuna</name>
      <value>6.54300</value>
    </domain>
    <domain>
      <name>storage</name>
      <value>2.39600</value>
    </domain>
    <domain>
      <name>nwk_legacy</name>
      <value>0.09000</value>
    </domain>
    <domain>
      <name>cifs</name>
      <value>0.00200</value>
    </domain>
  </domains>
</performance>
```


Refer to the Interoperability Matrix Tool (IMT) on the NetApp Support site to validate that the exact product and feature versions described in this document are supported for your specific environment. The NetApp IMT defines the product components and versions that can be used to construct configurations that are supported by NetApp. Specific results depend on each customer's installation in accordance with published specifications.